Perspectives on the scope and definition process of the Unified Enterprise Modelling Language*

Michaël Petit and Patrick Heymans

Computer Science Department, University of Namur, Rue Grandgagnage 21, B-5000 Namur, Belgium, {mpe,phe}@info.fundp.ac.be

Abstract. Interoperability of Enterprise Applications is a serious and multi-facetted problem. One of the tasks of the recently-started INTEROP Network of Excellence is to address this problem at the modelling level through the elaboration of a Unified Enterprise Modelling Language (UEML). In this paper, some methodological hints and an embryonic mission statement are submitted to discussion with peers.

1 Introduction

Today's "business trends are clearly towards the need for managing organizational and operational changes within companies in order to face global competition and fluctuating market conditions" [16]. This situation forces enterprises to tackle a number of integration and therefore interoperability problems: integration of markets, integration between several development and manufacturing sites, integration between suppliers and manufacturers, integration of design and manufacturing, integration of multi-vendor hardware and software components.

"Things to be integrated and coordinated need to be modelled. Thus, Enterprise Modelling (EM) is clearly a prerequisite for enterprise integration" [16]. However, the current status of the EM domain is characterized by a Tower of Babel situation in which many Enterprise Modelling Languages (EMLs) and tools are used. All these languages and tools have different (but sometimes similar) syntaxes, semantics, purposes and capabilities. Models of parts of the enterprise expressed in the different languages are therefore hard to understand, compare and combine. This situation hinders true enterprise integration, interoperability, and shared enterprise knowledge.

The recently-started INTEROP project [1] considers that one part of the solution is to define a common standard EML. Within its predecessor, the UEML project [2], the initial definition of the so-called Unified EML (UEML) version 1.0 was performed. The work included a requirements analysis and a subsequent definition of the language.

*This work is partially supported by the Commission of the European Communities under the sixth framework programme (INTEROP Network of Excellence, Contract 508011) [1]

Due to the nature of the UEML project[1], UEML 1.0 allows mainly the modelling of process aspects but leaves out other aspects (such as informational or organisational aspects) and covers only a small set of the identified requirements. It was defined by integrating three existing EMLs (namely subsets of GRAI [5], EEML [9] and IEM [11]) by following a methodology based on methodologies proposed in database integration [12].

Within INTEROP, UEML's expressiveness will be extended beyond process aspects and more requirements will be covered.

In this paper, basing on the experience in defining UEML and other modelling languages (i.e. the Albert language [3, 4]), we propose methodological guidelines and initial ideas for defining UEML 2.0.

2 Methodological hints for UEML extension

Three important inputs for the definition of UEML 2.0 are (i) UEML 1.0 itself, (ii) the list of UEML requirements and (iii) the methodology that was used for elaborating UEML on the basis of existing languages.

Due to the goals and duration of the UEML project explained above some of these results have limitations.

First, the requirements list, which was elaborated by collecting requirements from a large number of EM actors, suffers from some weaknesses:

- it is weakly structured;
- it mixes requirements of different levels of detail;
- it does not attach rationales to requirements;
- it does not offer a big picture of what UEML is about;
- it seems to address requirements that are beyond the "reasonable" scope of a UEML;
- it contains requirements that are not sufficiently clear or too complex and need further refinement.

Consequently, it may be difficult to base the UEML 2.0 requirements on this list of requirements and to get a consensus among the UEML designers and users on what requirements UEML 2.0 has to address.

Secondly, although UEML 1.0 is a powerful process modelling language, building it on only three languages is insufficient to claim that the language is a reference or standard. Other EMLs should be considered, compared and integrated into UEML. Currently, the three languages used come from the EM community. Since many enterprise languages also come from the Ontology community some of them should also be considered for integration e.g. [6, 15].

To address these problems, we propose the following more top-down methodology:

[1] Two characteristics of the UEML project explain these limitations: its short duration (15 months) and its objectives (investigating and demonstrating the feasibility of using UEML for exchanging models among three EM software environments).

1. Start by defining a *scope* document (mission statement) for UEML 2.0. This document would identify the main *problems to be solved* by UEML, the main *goals and non-goals* of UEML needed to solve these problems, and the main *alternative solutions* for elaborating an UEML (and their comparison with respect to the goals and non-goals), as well as the main *risks* of proposing a UEML (see Sect. 3);
2. Progressively refine top goals into more detailed requirements. This refinement should use some existing goal refinement methodology (such as those existing in software requirements engineering like i* [17]). The top goals would then be decomposed progressively into simpler goals until the requirements from UEML project or new requirements are identified.
3. Analyse and select existing EM languages and ontologies that currently best meet some of the requirements.
4. "Integrate" these languages and ontologies into UEML. For this, the methodology applied in the UEML project [12] can be enhanced and applied.

The methodology should also take advantage of the successes, difficulties and failures of similar initiatives (e.g. UML). Existing requirements documents written for similar purposes should also be considered (e.g. list of general requirements for programming languages [14], requirements list for PSL [13], ...).

In the following section, we discuss some initial ideas for the first step of the proposed methodology, namely, the scope definition. They are clearly all but complete and should be further discussed with the INTEROP partners before being elaborated.

3 Ideas for a UEML scope definition

3.1 Problems to be solved by UEML

The UEML should at least answer to the following two problems.

Problem 1. **Tower of Babel situation** (see Sect. 1).

Problem 2. **Scope of modelling unclear** Modelling of non-software artifacts is not common practice while it has potentially many benefits e.g. understanding and improving the enterprise and expressing assumptions on the environement of software [8]

3.2 UEML goals and non-goals

In order to solve these problems, we suggest some goals:

Goal 1. UEML should promote and improve EM practices.

Goal 2. UEML should provide effective and efficient means for EML users to achieve EM.

Goal 3. UEML should be considered the EM standard for EML users and providers.

Goal 4. UEML should minimize adaptation/transition efforts and costs of current enterprise EM users and providers.

Goal 5. UEML should foster the convergence of EML providers' contributions.

What should UEML be or not be:

Goal 6. UEML should be an EML, a coordinated set of EMLs or a family of related EMLs (e.g. through profiling). Based on the concept of language defined in [7], each such EML should have:

- one or more concrete syntaxes, that is (i) one or more user-oriented syntaxes (graphical and/or textual) and (ii) one or more tool-oriented syntaxes, including an exchange format.
- ONE abstract syntax aka meta-model (or several equivalent ones).
- ONE formally defined semantics (or several equivalent ones).

Goal 7. UEML should *not* be a mere exchange format (e.g. an XML-based syntax for exchanging enterprise models).

Goal 6 will help to solve both problems 1 and 2. Goal 7 is required because a mere exchange format (that is a tool-oriented language, hidden from the user) would not help to solve problem 1. This would not be sufficient to encourage the convergence of user-oriented languages.

UEML should additionally possess a number of qualities to insure its acceptance, which reduces problem 1 and 2:

Goal 8. UEML should be *expressive*: it should offer enough constructs to allow expressing all reasonably needed EM concepts.

Goal 9. UEML expressiveness should be *minimal*: it should not offer more constructs than those reasonably needed for EM.

Goal 10. UEML should be *natural*: it should allow to express EM concepts in a reasonably intuitive/natural way, that is, straightforwardly, without cumbersome "coding".

For more qualities see e.g. [10].

3.3 Risks associated to the definition of UEML

The definition of UEML as a standard EML could be compared e.g. to the definition of UML as a standard language for object-oriented modelling of software. The UML experience suggests the following risks for UEML's definition:

1. Definition of a unified language is a huge and tricky task;
2. The definition and evolution process of a language might endanger its quality. Important aspects to be defined are the composition of the consortium defining UEML and the decision protocol used;
3. Acceptance by the community is uncertain.

4 Conclusion

In this paper, we underline important methodological issues for a reference EML, one of the main tasks of the INTEROP project. We suggest to start the definition process of the so-called UEML by a mission statement aimed at driving future efforts. Parts of its content are proposed and submitted to discussion.

On the basis of this scope definition, candidate enterprise representation languages and ontologies will have to be selected for integration in UEML.

References

1. INTEROP project website. `http://www.interop-noe.org/`, April 2004.
2. UEML project website. `http://www.ueml.org`, April 2004.
3. Philippe Du Bois. *The Albert II Language : On the Design and The Use of a Formal Specification Language for Requirements Analysis.* PhD thesis, Computer Science Department, Facultés Universitaires Notre-Dame de la Paix, Namur, Belgium, September 1995.
4. Yves Bontemps and Patrick Heymans. Turning high-level live sequence charts into automata. In *Proc. of "Scenarios and State-Machines: models, algorithms and tools" (SCESM) workshop of the 24th Int. Conf. on Software Engineering (ICSE 2002)*, Orlando, FL, May 2002. ACM. `http://www.cs.tut.fi/~tsysta/ICSE/papers/`.
5. G. Doumeingts. *GRAI: Méthode de Conception Des Systèmes En Productique.* PhD thesis, University of Bordeaux I, France, 1984. in french.
6. Mark S. Fox and Michael Gruninger. Ontologies for enterprise integration. In Michael Brodie, Mathias Jarke, and Michael Papazoglou, editors, *Proc. of the Second International Conference on Cooperative Information Systems – CoopIS-94*, pages 82–89, Toronto (Canada), May 17-20, 1994.
7. D. Harel and B. Rumpe. Modeling languages: Syntax, semantics and all that stuff, part i: The basic stuff. Technical Report MCS00-16, Faculty of Mathematics and Computer Science, The Weizmann Institute of Science, 2000.
8. Michael Jackson. Some basic tenets of description. *Software and Systems Modelling Journal (Sosym)*, 1(1):5–9, 2002. `http://www.sosym.org/`.
9. H.D. Jorgensen and S. Carlsen. Emergent workflow: Integrated planning and performance of process instances. In *Proc. Of Workflow Management'99. Münster, Germany*, 1999.
10. J. Krogstie and A.Sølvberg. Information systems engineering: Conceptual modeling in a quality perspective. Technical report, NTNU, January 2 2000.
11. Kai Mertins and Roland Jochem. *Quality-Oriented Design of Business Processes.* Kluwer Academic Publishers, Boston/Dordrecht/London, 1999. ISBN 0-7923-8484-9.
12. Michaël Petit. Some methodological clues for defining a unified enterprise modelling language. In Kurt Kosanke, Roland Jochem, James G. Nell, and Angel Ortiz Bas, editors, *Enterprise Inter- and Intra-Organisational Intergration - Building an International Consensus.* Kluwer Academic Publishers, kurt kosanke, roland jochem, james g. nell and angel ortiz bas (editors) edition, 2003. ISBN 1-4020-7277-5.

13. Craig Schlenoff, Amy Knutilla, and Steven Ray. Unified process specification language: Requirements for modeling process. Technical Report NISTIR 5910, National Institute of Standards and Technology (NIST), Manufacturing Engineering Laboratory, Manufacturing Systems Integration Division, Gaithersburg, MD 20899, September 1996. `http://www.mel.nist.gov/msidlibrary/doc/schlen96/req-paper.pdf`.
14. Ulf Schünemann. Home-page of programming language design. `http://www.cs.mun.ca/~ulf/pld/`, April 2004.
15. Mike Uschold, Martin King, Stuart Moralee, and Yannis Zorgios. The enterprise ontology. *The Knowledge Engineering Review*, 13, 1998. Special Issue on Putting Ontologies to Use (eds. Mike Uschold and Austin Tate). Also available from AIAI as AIAI-TR-195, `http://www.aiai.ed.ac.uk/project/pub/documents/1998/98-ker-ent-ontology%.ps`.
16. François B. Vernadat. *Enterprise modeling and integration: principles and applications*. Chapman & Hall, 1996.
17. E. Yu. *Modelling Strategic Relationships for Process Reengineering*. Ph.D. Thesis, Univ. of Toronto, 1994.